REMARKS

In response to the final Official Action of August 21, 2007 and the Advisory Action of November 8, 2007, applicant has amended claims 1, 2, 17, 18, 19, 21-32, and 36 and has canceled claims 4 and 20 respectively in view of the amendment to independent claims 1 and 17.

This Amendment After Final is filed following the filing of a Notice of Appeal and Pre-Appeal Brief Request for Review on November 23, 2007 and the Notice of Panel Decision mailed on January 16, 2008. No extension of time is believed to be required.

Support for the amendment to claims 1, 17, and 36 is found in the application as originally filed, including Figures 1-3 and in the specification, including page 4, line 26 through page 6, line 12 and page 12, line 33 through page 16, line 28. No new matter is added.

Claim Rejections - 35 USC §103

At section 6, claims 1, 17, and 31-36 are rejected under 35 USC §103(a) as unpatentable over US patent 6,970,565, Rindsberg, further in view of US patent 7,149,901, Herbert, et al (hereinafter Herbert).

The Office asserts that Rindsberg teaches a method of enhancing data security having actions corresponding to claim 1, except for failing to specifically teach the generating of keys repeatedly. The Office then relies upon Herbert for teaching the generation of a new key repeatedly and using the new key to encrypt files to be stored. For the reasons set forth below, it is respectfully submitted that the combination of Rindsberg and Herbert does not suggest claim 1 as amended.

Amended Claim 1

Amended claim 1 presents what applicant considers to be a basic idea of the present invention as discussed in the specification as filed, including page 4, line 26 through page 6, line 12, as well as shown in Figures 1 and 2 and the

accompanying description thereof, including page 12, line 33 through page 16, line 28.

Thus, a basic idea of the present invention is that an electronic appliance (100) which includes a CPU (103) and a secure execution environment (104) is designed to provide for execution of applications that run in the appliance. The secure execution environment typically contains a read-only memory (104) that contains boot application software and an operating system, as well as a RAM (106) for the storage of data and applications. In order to run applications, it is necessary to read program code that is permanently stored outside of the secure execution environment, such as in flash memory (112). Such program code is typically outside of the secure execution environment due to the size of the program code. Such program code stored in memory (112) is typically strongly encrypted which makes it difficult to attack by a malicious user even if long periods of time are available for such an attack. However, such strongly encrypted program code is not readily executed by CPU (103) and the associated secure environment (104) since the strong encryption requires time-consuming resources to decrypt.

It is therefore much easier to read such program code by storing it in a memory (110), such as a random access memory, where the program code is less strongly encrypted. Such less strongly encrypted code can be executed by a processor (103) and the secure execution environment (104) at a sufficiently rapid rate so that the overall appliance (100) performs its intended functions in a timely fashion. The memory (110) is typically outside of the secure execution environment due to the size of the program code.

Since the program code stored in memory (110) is much more subject to malicious attack due to its less strong encryption, there is a problem with maintaining such code in the temporary memory unless the concept of the present invention is utilized; namely, to repeatedly encrypt the program code stored in memory (110) with a new secret key that is generated in the secure execution environment.

Thus, the present invention as set forth in amended claim 1, claims that in the secure execution environment, program code read from an external source where such program code is strongly encrypted is verified in the secure execution environment and that the verified code is then encrypted with a less strong key and that this less strongly encrypted program code is written into storage (memory) and that this set of actions is repeated; that is, that the less strong new secret key is again generated, the data is verified, encrypted, and stored in the temporary memory on a repeating basis.

As discussed in the specification, including page 5, lines 19-36, cryptoanalysis of program code is very difficult if the encryption key is repeatedly changed in the manner as disclosed in the present invention.

Thus, claim 1, as amended, recites the actions of reading strongly encrypted data external to a secure execution environment of an electronic device to which access is restricted, wherein said strongly encrypted data comprises program code to be executed in said electronic device.

The method of claim 1 further verifies in the secure execution environment the integrity of said strongly encrypted data, as well as generating in said secure execution environment a new secret key for less strongly encrypting said verified data. The method further recites less strongly encrypting in said secure execution device the verified data by means of said new secret key and writing the less strongly encrypted data into storage wherein at least some of said storage is external to said secure execution environment. The method further recites repeating the above-recited actions.

<u>The Cited Art</u>

Rindsberg is directed to a method for securely downloading and installing a program patch to a plurality of processing devices. The invention is particularly directed to situations where the program patch is transmitted over a communications channel that is particularly vulnerable to hacking (Rindsberg, column 6, lines 20-28). Initially, an encrypted program patch is transmitted to a device intended to receive the updated patch (Figure 3, step 91) where the patch

program is decrypted using a shared key (Figure 3, step 94). In Rindsberg, the integrity of the patch is checked (Figure 3, step 96) to determine if it is from the intended source or if errors are present and if it is not from the intended source or if errors are detected, the patch is deleted from memory (Figure 3, step 104). If the integrity check on the patch passes (Figure 3, step 98), the clear text patch program is re-encrypted using a unique key 63 corresponding to a unique ID 62 burned into the device at the time of manufacturing, where the unique key is known only to the processor (Figure 3, step 100) (see generally, Rindsberg, column 7, lines 36-39 and column 8, lines 4-33). The encrypted patch program is stored in data RAM (66). The contents of the patch program are then decrypted using the unique key and the clear text patch program is stored in RAM (74) (Rindsberg, claim 8, lines 37-56).

Herbert is directed to a method for maintaining integrity and confidentiality of pages that are paged to an external storage unit from a physically secure environment (Herbert, Abstract and Figure 1). An outgoing page is selected to be exported from a physically secure environment to an insecure environment. An integrity check value is generated, such as by the integrity check engine 13 so as to generate an integrity check value for later comparison when the page of data is subsequently paged back in. The outgoing page is then encrypted using a cryptographically strong encryption algorithm via encryption engine 12 based upon a random number generator 18 used to generate keying material for the encryption engine. By use of the encryption and the integrity check, the security of the data on the outgoing page can be maintained in an insecure environment since when it is read back, it is integrity checked with the previously generated integrity check value. If the integrity check is good, the page is allowed to populate a secure random access memory 14 within the physically secure environment 1 (see generally Herbert, column 2, line 47 through column 3, line 16).

Argument

As noted above, Rindsberg is directed to an apparatus and method for securely downloading and installing a program patch in a processing device. As shown in Figure 2 of Rindsberg and as discussed at column 6, line 54 through page 7, line 5, the program ROM (72) of processing device (60) is typically burned (i.e., programmed) when the device is manufactured. This program code is adapted to permit the device to perform all its required processing tasks. After manufacture, the device is imbedded in a product and released into the field. As noted in Rindsberg, it is typical that at some time after release into the field, bugs are found in the software stored in ROM (72) and it is therefore desirable to upgrade to software by means of a patch program. This patch program is typically stored in a patch RAM (74) in clear text.

The method disclosed in Rindsberg is for the downloaded patch to be decrypted (using the shared key from the host) and to then be encrypted using a unique key (63) known only to the device and corresponds to a particular, unique ID (62) of the device that is burned into the device during manufacture. This unique key is used to encrypt the patch program which is then stored in data RAM (66). As disclosed in Rindsberg, the clear text patch program is then stored in patch RAM (74) (step 118 of Figure 4).

As stated in Rindsberg, the purpose of using the unique key for re-encrypting the patch program originally received with the shared key is that the shared key is typically known to a large number of devices and is therefore more likely compromised and also, because the shared key can be changed at a relatively frequent rate. The patch program encrypted using a permanent unique key rather than the transitory shared key is stated as being more efficient to store in the device which is especially true since the patch program may be in service for relatively long periods of time (Rindsberg, column 8, line 68 through column 9, line 8).

Thus, Rindsberg does not disclose or suggest reading strongly encrypted data external to a secure execution environment of an electronic device. In the present invention, once the strongly encrypted data is verified, a new secret key

for less strongly encrypting said verified data is used to encrypt the verified data, followed by writing this less strongly encrypted data into storage, where at least some of the storage is external to the secure execution environment. This process is repeated in the present invention so as to insure that the less strongly encrypted data written into storage external to the secure execution device is less subject to malicious attack.

In Rindsberg, the patch program once verified is encrypted and stored in data RAM and its clear text version stored in patch RAM (74). This, of course, would be permissible since the disclosed device is shown as being secure relative to the outside environment. That is, there is no memory outside of the device comprising data, including program code for execution in said electronic device.

Therefore, even if there was some reason to repetitively update the unique key in Rindsberg, there is no showing in Rindsberg that this unique key is less strongly encrypted than the shared key used by the external host nor that the less strongly encrypted data is written into storage, where at least some of the storage is external to the secure execution environment, wherein at least some of the data comprises program code to be executed in the electronic device.

Furthermore, it is specifically disclosed in Rindsberg that the unique key 63 used to encrypt the patch program is a key which is only known to the device itself and typically each such key corresponds to a particular unique ID 62 which is burned into the device during manufacture (Rindsberg, column 7, lines 36-39). If, as proposed by the Office this permanent unique key is changed, the device in Rindsberg may no longer be identified as having that particular unique ID. Furthermore, Rindsberg specifically claims the idea that the unique key is derived from a unique ID that is <u>permanently</u> burnt into the computing device (see, for example, claim 36) and thus there is no suggestion in the disclosure or claims of Rindsberg toward having a repeatingly altered unique key. This is in contradistinction to the argument presented at section 4 of the final Official Action where the Office asserts that a person of ordinary skill in the art would have

combined Rindsberg and Herbert to modify the unique key of Rindsberg repeatedly.

The Office argues in the Advisory Action that a rotating key in Rindsberg would not change its operation since it could still be used to identify the device. But if this were so, then clearly Rindsberg would not teach burning the ID (and therefore the unique key) into the device at the time of manufacture.

In summary, the unique ID in Rindsberg has the functionality of identifying the device and the unique key is permanently derived therefrom. To repeatedly change that unique key would in effect negate this functionality (see Abstract of Rindsberg).

In fact MPEP §2143.01 VI addresses this situation.

> "VI The Proposed Modification Cannot Change the Principle of Operation of a Reference
>
> If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facia* obvious. *In re Ratti,* 270 F.2d 810, 123 USPQ 349 (CCPA 1959)."

Here, as in the cited *In re Ratti* decision, to modify the unique key in Rindsberg which is based on a unique ID of the device with a repeating new key would effectively negate the identification property of the unique key disclosed in Rindsberg.

In short, a person of ordinary skill in the art, would not under any normal circumstances be motivated to change unique key of Rindsberg repeatedly. Consequently, the particular combination asserted by the Office of using Herbert's key generation with the unique key disclosed in Rindsberg would not in fact be possible or obvious to a person of ordinary skill in the art at the time of the present invention.

It is therefore respectfully submitted that Rindsberg fails to suggest the need or desire for repeatedly generating a new secret key that is used to less strongly encrypt in a secure execution environment verified data and writing this less

strongly encrypted data into storage where at least some of said storage is external to said secure execution environment.

For all of the foregoing reasons, it is therefore respectfully submitted that claim 1, as amended, is not unpatentable under 35 USC §103 in view of Rindsberg, further in view of Herbert.

Due to amendment of claim 1, claim 4 has been canceled.

Independent apparatus claims 17 and 36 have been amended in a manner similar to method claim 1 and are, therefore, also believed to be distinguished over Rindsberg in view of Herbert. Claims 17-36 now recite an apparatus since the system referred to in the specification is also referred to as a device (see specification, page 12, lines 22-32).

Claim 20 has been canceled in view of the amendment to claim 17.

Since each of the independent claims of the present application is believed to be distinguished over the cited art, it is respectfully submitted that claims 2, 3, 5-16, 18, 19, and 21-35 are further distinguished over the cited art at least in view of such dependency.

It is therefore respectfully submitted that the present application as amended is in condition for allowance and such action is earnestly solicited.

Respectfully submitted,

Dated: February 19, 2008

Alfred A. Fressola
Attorney for Applicant
Registration No. 27,550

WARE, FRESSOLA, VAN DER SLUYS
& ADOLPHSON LLP
Bradford Green, Building Five
755 Main Street, P.O. Box 224
Monroe, CT 06468
Telephone: (203) 261-1234
Facsimile: (203) 261-5676
USPTO Customer No. 004955